California
Institute of
Technology

Adapting ODC for Empirical Evaluation of Pre-Launch Anomalies

Dr. Robyn Lutz and Carmen Mikulski

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The work was sponsored by the NASA Office of Safety and Mission Assurance under the Software Assurance Research Program led by the NASA Software IV&V Facility. This activity is managed locally at JPL through the Assurance and Technology Program Office

OSMA Software Assurance Symposium

July 29-August 1, 2003

California
Institute of
Technology

Topics

- **Overview**
- **Results**
  - **Mars Exploration Rover**
  - **Deep Impact**
  - **Infusion of method**
  - **Dissemination of results**
- **Examples: Patterns and Lessons Learned**
- **Benefits**

Overview

California Institute of Technology

- **Goal:**
  1. **To characterize pre-launch software anomalies, using data from multiple spacecraft projects, by means of a defect-analysis technology called *Orthogonal Defect Classification (ODC).***
  2. **To support transfer of ODC to NASA projects through applications and demonstrations.**
- **Approach:**
  1. **Analyze anomaly data using adaptation of *Orthogonal Defect Classification (ODC)* method**
     - Developed at IBM; widely used by industry
     - Quantitative approach
     - Used here to detect patterns in anomaly data
     - More information at http://www.research.ibm.com/softeng
  2. **Adapt ODC for NASA use and apply to NASA projects**

Overview: *Status*

California
Institute of
Technology

- **Previous work used ODC to analyze safety-critical *post-launch* software anomalies on 7 spacecraft.**
- **FY'03 task extends ODC work to *pre-launch development and testing* (Mars Exploration Rover testing, Deep Impact, contractor-developed software) and to support *technology infusion***
- **Adapted ODC categories to spacecraft software at JPL:**
  - **Activity: what was taking place when anomaly occurred?**
  - **Trigger: what was the catalyst?**
  - **Target: what was fixed?**
  - **Type: what kind of fix was done?**

Results: *MER*

California
Institute of
Technology

- **Collaborating with Mars Exploration Rover to experimentally extend ODC approach to pre-launch software problem/failure testing reports (525 to date)**
  - **Adjusted ODC classifications to testing phases**
  - **Institutional defect database → Access database of data of interest → Excel spreadsheet with ODC categories → Pivot tables with multiple views of data**
  - **Frequency counts of Activity, Trigger, Target, Type, Trigger within Activity, Type within Target, etc.**
  - **User-selectable representation of results support tracking trends and progress:**
    - **Graphical summaries**
    - **Comparisons of testing phases**
  - **Provides rapid quantification of data**
  - **Project provides feedback/queries on our monthly deliverables of results and on our draft reports/paper**

Results: *Deep Impact*

- **Collaborating with Deep Impact to extend ODC approach into software developmental anomaly reports via ODC classification of development-phase SCRs (Software Change Re[redacted]rts) at Ball**
  - **Classified initial set of 94 critical DI SCRs (with highest cause-corrective action/failure effect ratings)**
  - **Feasibility check: ODC classification of development-stage software defects works well**
  - **Initial delivery to DI of ODC pivot table results (for browsing), of user instructions, and of initial issues/concerns**
  - **Project uses telecons/email to answer questions, suggest paths of interest to project**

Results: *Infusion*

California
Institute of
Technology

- **Gave invited presentation on ODC to JPL's Software Quality Initiative task as candidate defect-analysis tool**
- **Worked with manager/developers of next-generation JPL problem-reporting system to ensure that their web-based database will support projects' choice of ODC**
- **Carmen presented ODC to JPL's DII project (Defense Information Infrastructure); they requested followup presentation (given); Carmen is working with DII as they train their users & transition ODC into their operations**
- **Wide distribution to projects of slide summarizing use of ODC on MER (T. Menzies suggestion)**

Use on MER of ODC* defect analysis

Testing Problem Reports

Problem Report file for MER

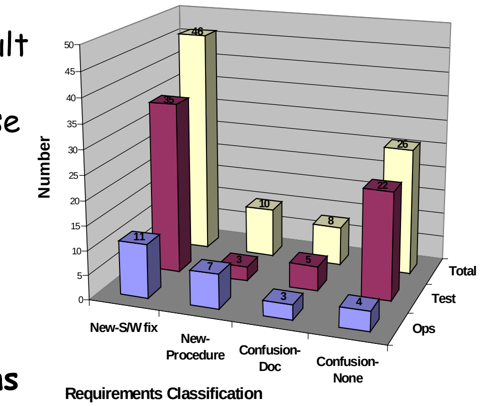
Classify MER test problem reports per ODC Categories

➢Where are the spikes?
➢Phase-by-phase deltas?
➢Activity/Trigger/Target/Type look nominal?

MER use: Browse pivot chart (Excel) for overview/closer look at testing

Discover defect patterns in testing of interest to MER

➢How many problem reports result in requirements discovery?
➢Improvement release-by-release uneven: why?
➢Many closed with no fix: why?



MER use: Identify patterns of concern for more investigation

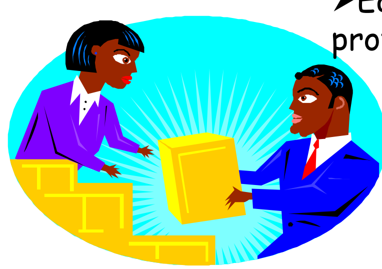
Perform Root Cause Analysis on subset

MER use: Improved understanding of data, underlying causes, defect mechanisms

• Develop and package recommendations for MER

Recommendations for MER and future projects:
➢Earlier assignment of criticality ratings
➢If software's behavior confused testers, enhance documentation
➢Earlier testing of fault-protection

MER use: Implement/defer recommendations



*ODC = Orthogonal Defect Classification technique [IBM]

California
Institute of
Technology

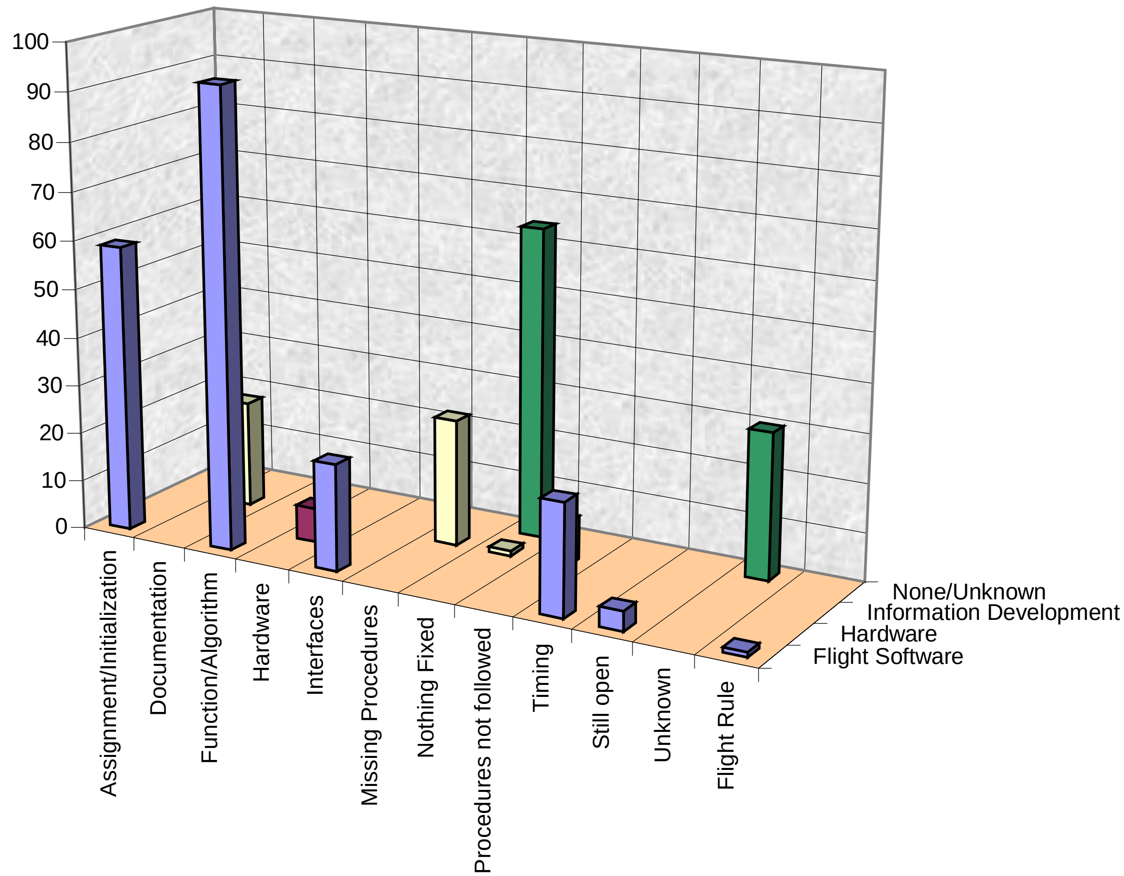
Results: *Dissemination*

- **Presented paper on the 4 mechanisms involved in requirements discovery during testing at ICSE 2003 (Int'l Conf on S/W Eng)**
- **Presented paper on patterns of defect data at SMC-IT 2003 (Space Mission Challenges)**
- **Presented results at JPL/GSFC QMSW 2003 (Quality Mission Software Workshop)**
- **T. Menzies presented analysis of the ODC defect data at SEKE 2003 (Int'l Conf S/W Eng & Knowledge Eng)**
- **Paper on how operational anomalies drive requirements evolution appeared in Journal of Systems and Software, Feb. 2003**
- **Paper describing similar mechanisms in testing & ops anomalies accepted to RE 2003 (Int'l Requirements Eng Conf); selected as one of best experience papers & paper invited for IEEE Software submission**

Example: Testing Defect Patterns

California
Institute of
Technology

Distribution of Types by Target



Example: Lessons Learned from ODC

California Institute of Technology

- **Testing reports give "crystal ball" into operations**
  - **False-positive testing problem reports (where software behavior is correct but unexpected) provide insights into requirements confusions on the part of users**
  - **If software behavior surprised testers, it may surprise operators**
- **Closing problem reports with "No-Fix-Needed" decision can waste opportunity to document /train/ change procedure**
  - **Avoid potentially hazardous recurrence**
  - **Important in long-lived systems with turnover, loss of knowledge**

Benefits

California Institute of Technology

- **Experience: Applied to 9 NASA projects**
  - **Development, testing, and operations phases**
- **Level of effort affordable**
  - **Uses existing fields in existing problem-reporting system)**
  - **ODC ~ 4 minutes/defect vs. Root cause ~ 19 (Leszak & Perry 2003)**
- **Uses metrics information to identify and focus on problem patterns**
  - **Incorporates project results into multi-project baseline patterns to provide guidance to future projects**
  - **Can answer current project's questions regarding defects**
- **Flexible**
  - **Visualization & browsing options**
- **Provides quantitative foundation for process improvement**
- **Equips us with a methodology to continue to learn as projects and processes evolve**

California
Institute of
Technology

Backup Slides

Results

California Institute of Technology

- **2 basic kinds of requirements discovery:**
  - **Discovery of *new* (previously unrecognized) requirements or requirements knowledge**
  - **Discovery of *misunderstandings* of (existing) requirements**
- **Reflected in ODC Target (what gets fixed) and ODC Type (nature of the fix):**
  1. Software change **(new requirement allocated to software)**
  2. Procedural change **(new requirement allocated to operational procedure)**
  3. Document change **(requirements confusion addressed via improved documentation)**
  4. No change needed

Results: Examples

1. **Incomplete requirements, resolved by change to software:**

   New software requirement became evident: initial state of a component's state machine must wait for the associated motor's initial move to complete

2. **Unexpected requirements interaction, resolved by changes to operational procedures:**

   Software fault monitor issued redundant off commands from a particular state (correct but undesirable behavior). Corrective action was to prevent redundant commands procedurally by selecting limits that avoid that state in operations

Results: Examples

3. Requirements confusion, resolved by changes to documentation
   - Testing personnel incorrectly thought heaters would stay on as software transitioned from pre-separation to Entry/Descent mode; clarified in documentation.
4. Requirements confusion, resolved without change
   - Testers assumed commands issued when component was off would be rejected, but commands executed upon reboot. No fix needed; behavior correct.